

Detecting the Change of Clustering Structure in Categorical Data Streams

Keke Chen

Ling Liu

Georgia Institute of Technology
{kekechen, lingliu}@cc.gatech.edu

Abstract

Clustering data streams can provide critical information for making decision in real-time. We argue that detecting the change of clustering structure in the data streams can be beneficial to many real-time monitoring applications. In this paper, we present a framework for detecting changes of clustering structure in *categorical data streams*. The change of clustering structure is detected by the change of *the best number of clusters* in the data stream. The framework consists of two main components: the BkPlot method for determining the best number of clusters in a categorical dataset, and the summarization structure, Hierarchical Entropy Tree (HE-Tree), for efficiently capturing the entropy property of the categorical data streams. HE-Tree enables us to quickly and precisely draw the clustering information from the data stream that is needed by BkPlot method to identify the change of best number of clusters. Combining the snapshots of the HE-Tree information and the BkPlot method, we are able to observe the change of clustering structure online. The experiments show that HE-Tree + BkPlot method can efficiently and precisely detect the change of clustering structure in categorical data streams.

1 Introduction

With the deployment of wide-area sensor systems and Internet-based continuous query applications, processing stream data has become a critical task. As an important method in data analysis, recently clustering has attracted more and more attention in analyzing and monitoring streaming data [15, 1]. The initial research has shown that clustering stream data can provide important clues about the new emerging data patterns so that the decision makers can predict the coming events and react in near real time. Stream data clustering is especially important to the time-critical areas such as disaster monitoring, anti-terrorism, and network intrusion detection. As many of such applications also include a large amount of categorical data, clustering the categorical data streams becomes interesting and challenging problem. However, very few [3] have addressed the problems related to clustering categorical data streams. Moreover, no one addressed how to effectively monitor the change of clustering structure in categorical data streams.

The change of clustering structure in data streams involves three aspects: the increasing size of clusters, new emerging clusters, and disappearing clusters, which might be caused by the mixing of two growing clusters or the increasing noise level. The change of clustering structure may correlate to some important events in the applications, for example, a network attack happening with the emerging of new clusters. In this context, we are more interested in detecting both the emerging of clusters and the disappearing of clusters. The former may indicate some new events are going to happen, and the later implies two possible situations: 1) the previously identified clusters are false clusters – they should be a part of outliers; 2) some clusters grow big and become merged, which may indicate that an existing event will be occurring more and more frequently.

Detecting the change of clustering structure for categorical data streams is a complicated problem. As the first step solution towards this goal, we propose a method for *identifying the change of the number of clusters in categorical data streams*, so that the emerging of new clusters and disappearing of old clusters can be precisely detected. The basic idea involves two points.

1. Correctly identifying the best “K” for categorical clustering over any specific segment of data stream. Only when we can precisely identify the best number of clusters in a data stream, can we detect the change of the number of clusters. We have developed the BkPlot method working in general categorical clustering in [9] for identifying the best “K”. The first challenge for us is the need to extend and enable the BkPlot algorithm to work in the data stream environment.
2. The techniques supporting the precise summarization of the categorical data streams so that the BkPlot method can be applied to the data summarization, which is the focus of this paper. We design and implement a tree based structure – *Hierarchical Entropy Tree*, **HE-Tree** for short, to efficiently sketch the entropy characteristics of the categorical data stream and to support the best “K” monitoring.

Entropy-based Categorical Clustering Clustering techniques for categorical data are very different from those for numerical data, mainly because of the definition of similarity measure. Most numerical clustering techniques have been using distance functions, for example, Euclidean distance, to define the similarity measure. However, there is no such inherent distance meaning between the categorical values.

In contrast to the distance-based similarity measure for pairs of data records, similarity measures based on the “purity” of a bulk of records seem more intuitive for categorical data. Entropy [11] is a well defined measure for the purity of dataset. Originally from information theory, entropy has been applied in various areas, such as pattern discovery [6], numerical clustering [10] and information retrieval [24]. Due to the lack of intuitive distance definition for categorical values, recently entropy has been applied in clustering categorical data [3, 21, 8, 12]. The initial results have shown that the entropy criterion can be very effective in clustering categorical data. Li et al [21] also proved that the entropy criterion can be formally derived in the framework of probabilistic clustering models, which strongly supports that the entropy criterion is a meaningful and reliable similarity measure, particularly good for categorical data.

In entropy-based categorical clustering, the quality of clustering is essentially evaluated by the entropy criterion, namely, the *Expected Entropy* of clusters [3, 21]. Other variants, such as Minimum Description Length (MDL) [8] or mutual information [12, 2], turn out to be equivalent to the entropy criterion, as Li et al [21] shows. We categorize all these approaches as entropy-based categorical clustering. The main goal of the entropy-based clustering algorithms is to find a partition that minimizes the expected entropy for K clusters. However, minimizing expected entropy is a NP-hard problem, thus it is computationally intractable even for a median-size dataset. A common approach to solving this problem is approximation. Typically, in approximation algorithms we have to sacrifice some optimality to obtain the efficiency. Streaming categorical data makes it even harder to balance the two conflicting factors. **Monitoring the Change of Clustering Structure with Entropy-criterion** Existing entropy-based categorical clustering algorithms are not well balanced between the optimality and the efficiency, thus not working for monitoring the changes in categorical data stream. For example, although Coolcat [3] aims at processing streaming data, it is very likely to trap in local minima, and thus cannot ensure precise detection of the inherent clustering structure [9]. While other methods, such as Monte-Carlo method [21] and Cross Association method [8] can give results in better quality, it is not easy to extend them to process data streams.

In this paper, we propose an entropy-based approach for monitoring the change of clustering structure in categorical data streams. The approach extends the idea of BkPlot [9] for identifying the “best K ” for entropy-based categorical clustering, to monitor the change of clustering structure in data streams. The key of this approach is the HE-Tree summarization structure, which can efficiently preserve the entropy characteristics of the streaming data. Our experiments show that HE-Tree provides high-quality information for BkPlot method to precisely identify the change of clustering structure.

The rest of the paper is organized as follows. Section 2 sets down the notations and gives the definition of the traditional entropy-based clustering criterion. Section 3 briefly describes the BkPlot designed for general detection of “best K ” in categorical clustering. In section 4, we develop the HE-Tree structure and describe its working mechanism. The framework of the approach is summarized in section 5. The experimental results are shown in section 6. We review the related categorical clustering work and stream

clustering in section 6, and conclude our work in section 7.

2 Notations and Definitions

We first give the notations used in this paper and then introduce the traditional entropy-based clustering criterion. Several basic properties about the entropy criterion will be presented later.

Consider that a dataset \mathbb{S} with N records and d columns, is a sample set of the discrete random vector $X = (x_1, x_2, \dots, x_d)$. For each component x_j , $1 \leq j \leq d$, x_j takes a value from the domain A_j . A_j is conceptually different from A_k ($k \neq j$). There are a finite number of distinct categorical values in $\text{domain}(A_j)$ and we denote the number of distinct values as $|A_j|$. Let $p(x_j = v)$, $v \in A_j$, represent the probability of $x_j = v$, we have the classical entropy definition [11] as follows.

$$\begin{aligned} H(X) &= \sum_{j=1}^d H(x_j) \\ &= - \sum_{j=1}^d \sum_{v \in A_j} p(x_j = v) \log_2 p(x_j = v) \end{aligned}$$

Since $H(X)$ is estimated with the sample set \mathbb{S} , we define the estimated entropy as $\hat{H}(X) = H(X|\mathbb{S})$, i.e.

$$\begin{aligned} \hat{H}(X) &= H(X|\mathbb{S}) \\ &= - \sum_{j=1}^d \sum_{v \in A_j} p(x_j = v|\mathbb{S}) \log_2 p(x_j = v|\mathbb{S}) \end{aligned}$$

Suppose the dataset \mathbb{S} is partitioned into K clusters. Let $C^K = \{C_1, \dots, C_K\}$ represent a partition, where C_k is a cluster and n_k represent the number of records in C_k . The classical entropy-based clustering criterion tries to find the optimal partition, C^K , which maximizes the following entropy criterion [5, 7, 21].

$$O(C^K) = \frac{1}{d} \left(\hat{H}(X) - \frac{1}{N} \sum_{k=1}^K n_k \hat{H}(C_k) \right)$$

Since $\hat{H}(X)$ is fixed for a given dataset \mathbb{S} , maximizing $O(C^K)$ is equivalent to minimize the item $\frac{1}{dN} \sum_{k=1}^K n_k \hat{H}(C_k)$, which is named as the “expected entropy” of partition C^K . Let us notate it as $\bar{H}(C^K)$. For convenience, we also name the varying part $n_k \hat{H}(C_k)$ as the “Weighted Entropy” of cluster C_k .

Entropy criterion is especially good for categorical clustering due to the lack of intuitive distance function for categorical values. While entropy criterion can also be applied to numerical data [10], it is not the best choice since it cannot describe the cluster shapes and other specific clustering features for the numerical data.

3 BkPlot for Determining the “Best K ” for Categorical Clustering

Traditionally, statistical validity indices based on geometry and density distribution are applied in clustering numerical data [17]. A typical index curve consists of the statistical index values for different K number of clusters. The K s at the peaks, valleys, or distinguished “knees” on the index

curve, are regarded as the candidates of the optimal number of clusters (the best K). BkPlot method tries to find such kind of index for categorical data.

The basic idea in [9] is to investigate the entropy difference between any two neighboring partitions. The neighboring partitions are defined as two clustering results having K and $K + 1$ number of clusters, respectively. We notate their expected-entropy as $\bar{H}(C^K)$ and $\bar{H}(C^{K+1})$.

The curve of optimal $\bar{H}_{opt}(C^K) = \min\{\bar{H}_i(C^K)\}$, where i is the index of all possible K -cluster partitions, was identified as a smoothly decreasing curve without any distinguished peaks, valley, or knees, which we cannot effectively identify the best K .

However, the special meaning behind the difference of the neighboring partitions suggests us to explore the best K on the entropy difference between the neighboring partitions. Let the increasing rate of entropy between the optimal neighboring partitions defined as $I(K) = \bar{H}_{opt}(C^K) - \bar{H}_{opt}(C^{K+1})$. We identify that $I(K)$ implies two levels of difference between the neighboring partitions.

- $I(K)$ is the level of difference between the two neighboring schemes. The larger the difference, the more significant the clustering structure is changed by reducing the number of clusters by 1.
- Consider $I(K)$ as the amount of impurity introduced from $K + 1$ -cluster scheme to K -cluster scheme. If $I(K) \approx I(K + 1)$, i.e. K -cluster scheme introduces similar amount of impurity as $K+1$ -cluster scheme does, the change of clustering structure follows the “similar pattern, thus we can also consider that there is no significant difference from $K+2$ -cluster partition to K -cluster partition.

We define the differential of expected-entropy curve as “Entropy Characteristic Graph (ECG)” (Figure 2). An ECG shows that the similar partition schemes with different K are at the same “plateau”. From plateau to plateau there are the critical points implying the significant change of clustering structure, which could be the candidates for the best K s.

The common way to automatically identify such critical knees on ECG is to find the peaks/valleys at the second-order differential of ECG. Since an ECG consists of a set of discrete points, we define the second-order differential of ECG as $\delta^2 I(K) : \delta I(K) = I(K) - I(K + 1)$ and $\delta^2 I(K) = \delta I(K - 1) - \delta I(K)$ to make K aligned with the critical points. These critical points are highlighted in the second-order differential of ECG, which is named as “Best-K Plot (BkPlot)”.

Exact BkPlots cannot be achieved in practice, since $I(K)$ is based on the optimal K -cluster scheme which involves entropy minimization. However, since we only pay attention to the peak/valley points, approximate but accurate BkPlots are possible to acquire. A hierarchical clustering algorithm ACE in [9] is proposed to generate such BkPlots, and we have shown in experiments that ACE is a robust method to generating the high-quality BkPlots. ACE also has a nice property that we only need to observe the peaks in the BkPlots generated by ACE to determine the best K s. For interested readers, please find more details in [9].

ACE can be potentially applied to monitor the change of clustering structure in categorical data stream. However,

ACE is initially designed for static datasets. The $O(N^2)$ complexity prevents it working directly on data streams. In the next section, we will design an entropy based data summarization technique for categorical data streams, the result of which can be combined with the ACE algorithm to generate high-quality BkPlots for data streams.

4 HE-Tree: Capture the Entropy Characteristics of Categorical Data Stream

The basic idea of HE-Tree is to coarsely but rapidly assign the records from the data stream onto hundreds or thousands subclusters. Observing these hundreds or thousands subclusters will give us a precise estimation of the clustering structure. HE-Tree determines the subclusters only based on the previously processed data records, and the structures of the subclusters can automatically adapt to the new coming data records. The criterion for forming a subcluster is minimizing the expected-entropy of the subclusters – certainly, this minimization is only locally optimal. To fast locate a subcluster for a coming new record, we organize the subclusters in a tree, i.e., HE-Tree.

A HE-Tree consists of two key components:

1. *incremental entropy based similarity*, which helps to locate the branches for a new coming record;
2. *summary table*, which summarize the entropy characteristics of a subcluster

Given the fixed height h and fanout f , HE-Tree is constructed in two stages:

1. *growing stage*, which happens at the beginning of processing data stream;
2. *absorbing stage*, which absorb the new coming items to the subclusters at the leaf nodes, when the tree is full.

We first present incremental-entropy based similarity measure that is used in HE-Tree construction, and then give the structure of the HE-Tree node, which includes the summary table and other structure for fast entropy calculation. After that, we will focus on the construction algorithms of the HE-Tree.

4.1 Incremental Entropy

Incremental entropy is a tool used to describe the similarity between any two clusters. We begin with the observation about merging two clusters. Intuitively, merging the two clusters that are similar in the inherent structure will not increase the disorderliness (expected-entropy) of the partition, while merging dissimilar ones will inevitably bring larger disorderliness. Therefore, this increase of expected entropy has some correlation with the similarity between clusters. It is, thus, necessary to formally explore the property of merging clusters. Let $C_p \cup C_q$ represent the mergence of two clusters C_p and C_q , and C_p and C_q have n_p and n_q members, respectively. Suppose that the number of clusters is $K+1$ before the mergence happens. By the definition of expected entropy, the difference between $N\hat{H}(K)$ and $N\hat{H}(K + 1)$ is only the difference between the weighted entropies, $(n_p + n_q)\hat{H}(C_p \cup C_q)$ and $n_p\hat{H}(C_p) + n_q\hat{H}(C_q)$. Intuitively, since merging always increases the entropy, we have the following relation for the weighted entropies [9].

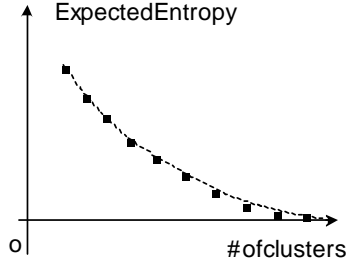


Figure 1: Sketch of expected entropy curve.

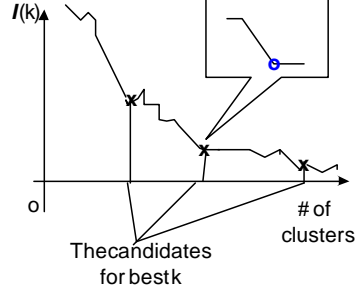


Figure 2: Sketch of ECG graph.

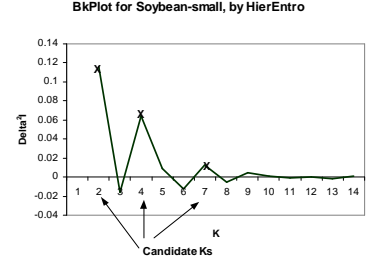


Figure 3: Finding the best k with BkPlot.

Proposition 1. $(n_p + n_q)\hat{H}(C_p \cup C_q) \geq n_p\hat{H}(C_p) + n_q\hat{H}(C_q)$

We name $I_m(C_p, C_q) = (n_p + n_q)\hat{H}(C_p \cup C_q) - (n_p\hat{H}(C_p) + n_q\hat{H}(C_q)) \geq 0$ as the “Incremental Entropy” of merging the clusters C_p and C_q . Note that $I_m(C_p, C_q) = 0$ suggests that the two clusters most likely have the identical structure – for every categorical value v_i in any arbitrary attribute x_j , $1 \leq i \leq |A_j|$, $1 \leq j \leq d$, we have $p(x_j = v_i|C_p) = p(x_j = v_i|C_q)$. The larger I_m , the more different the two clusters are.

4.2 Structure of HE-Tree

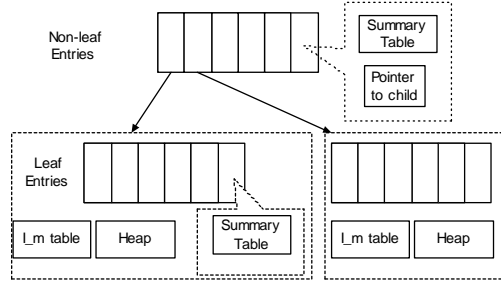


Figure 4: The structure of HE-Tree.

Summary Table. Summary table is used to maintain the fast calculation of the entropy $\hat{H}(C_k)$ and each node in the HE-Tree maintains one summary table (Figure 5). Since computing cluster entropy is based on counting the occurrences of categorical values in each column, summary table is used to keep the counters for each cluster. If the average column cardinality is m , a summary table keeps dm counters.

Summary table also has the important property.

Observation 1. When two clusters are merged, the sum of the two summary tables becomes the summary table for the new cluster.

Nodes in HE-Tree. HE-Tree is a balanced tree similar to B-tree, where each node has f entries and the entries in the leaf nodes represents the n_c subclusters. As shown in Figure 4, each entry in leaf node contains a summary table, and a leaf node also contains a I_m table with $(f+1)^2$ entries and a heap in size f for fast locating and merging the entries. I_m table keeps the value $I_m(i, j)$ for any pair of entries. Together with the heap, it is fast to keep track of the minimum I_m . An internal node (non-leaf) in the tree contains only the aggregation information of its child nodes – each entry in the internal node points to a child node and

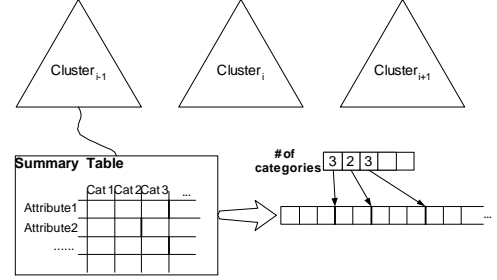


Figure 5: The structure of summary table.

its summary table is the sum of the summary tables in the child node.

In short, a HE-Tree is a “summary table tree”. Let a summary table represented as a vector \vec{s} and the entropy characteristic of any internal node C_i denoted as $EC_i(n_i, \vec{s}_i)$, where n_i is the number of records summarized by this node. Let C_{ij} , $1 \leq j \leq f$ represent the child nodes of C_i . HE-Tree maintains the property

$$EC_i(n, \vec{s}) = \sum_{j=1}^f EC_{ij}(n_{ij}, \vec{s}_{ij}) = EC_i\left(\sum_{j=1}^f n_{ij}, \sum_{j=1}^f \vec{s}_{ij}\right)$$

and tries to minimize the expected entropy $\bar{H}(C_i^f)$ in each insertion of new record. This local minimization is achieved through the algorithms in constructing the HE-Tree.

4.3 Constructing HE-Tree

The construction of HE-Tree consists of two phases: *the growing phase* and *the absorbing phase*. The algorithms for construction are carefully designed to minimize the expected entropy of the subclusters and adapt to the change of entropy in data stream.

Growing Phase. In the growing phase, the tree grows until the number of leaf nodes reaches $\lceil n_c/f \rceil$. When a new coming record is inserted into the existing tree, the first subroutine is for locating the target leaf node for insertion/absorption. Let e denote the inserted record and e_i denote one of the entry in current node. The search begins at the root node. Since each entry in the internal node is the summarization of its sub-tree, we can find the most similar entry to e by finding the minimum I_m among $I_m(e, e_i)$, $i = 1..f$, i.e.

$$e_t = \operatorname{argmin}_{e_i} \{I_m(e, e_i), i = 1..f\}$$

Iteratively, the same criterion is applied to the selected child node until a leaf node is reached. If the target leaf node has

empty entries and $I_m(e, e_i) \neq 0$ for all occupied entries, the record occupies one empty entry. We give the sketch of the subroutines in Algorithms 1 and 2.

Algorithm 1 HE-Tree.locate(node, e)

```

node ← target node, e ← target entry
if node is leaf then
    return node
end if
for Each entry  $e_i$  in node do
     $I_m^i \leftarrow I_m(e, e_i)$ 
end for
 $e_t \leftarrow \operatorname{argmin}_{e_i} \{I_m^i\}$ 
return locate( $e_t$ .subtree, e)

```

Algorithm 2 HE-Tree.insert(node, e)

```

e ← inserted entry, node ← target node
for Each entry  $e_i$  in node do
    if  $I_m(e, e_i) == 0$  then
        merge(e,  $e_i$ ), return
    end if
end for
if node.have_empty_entry() then
    node.enter(e)
    if (node.num_entry() ==  $f - 1$  and (not tree_full() or is_internal(node))) then
        split(node)
    end if
else
    micro-merging(node, e) //2nd phase
end if

```

When the target leaf node is full, a split operation is applied. In split algorithm, we partition the entries into two groups. First, two pivot entries (e_r, e_s) is found in the target node that have the maximum I_m if merging them – they are regarded as the most dissimilar pair among all pairs.

$$(e_r, e_s) = \operatorname{argmax}_{e_r, e_s} \{I_m(e_r, e_s), i = 1..f\}$$

The two pivot entries then become the two seed clusters. The rest entries are sequentially assigned to the two clusters so that the overall entropy of the partition keeps minimized. A new node is generated accommodating one of the two sets of entries. At the same time, one entry is added into the parent node pointing to the new node. The insertion/splitting continues until the number of leaf entries reaches n_c . Algorithm 3 gives the detailed description.

Absorbing Phase. In the second phase, the same locating algorithm is applied to locate the target leaf node for the new record. However, we have no insertion allowed since the entries are all occupied. Instead, in the leaf node we merge the most similar two items among the $f+1$ items – the f entries in the leaf node plus the new record. This allows the tree to rapidly adapt to the change of clustering structure in the entry level. In each leaf node, we maintain a I_m table and a heap for the f entries. When a new record comes, only f calculations of incremental entropy are needed to update the I_m table and the heap, before selecting the most similar two. Since this mergence is done within the f members, the cost is small, namely, about $O(dm f)$ for each record.

Algorithm 3 HE-Tree.split(node)

```

node ← target node
( $e_a, e_b$ ) ←  $\operatorname{argmax}_{(e_i, e_j)} \{I_m(e_i, e_j)\}$ 
 $\text{partition}_a \leftarrow e_a, \text{partition}_b \leftarrow e_b$ 
for Each entry  $e_i$  in node do
    if  $I_m(\text{partition}_a, e_i) < I_m(\text{partition}_b, e_i)$  then
         $\text{partition}_a \leftarrow \text{partition}_a \cup e_i$ 
    else
         $\text{partition}_b \leftarrow \text{partition}_b \cup e_i$ 
    end if
end for
if is_leaf(node) and not done then
    re-insert(root, entries in  $\text{partition}_a$ )
else
    newnode ←  $\text{partition}_a$ , remove(node,  $\text{partition}_a$ )
     $e_{\text{new}} \leftarrow \text{summary}(\text{newnode})$ , insert(node.parent,  $e_{\text{new}}$ )
end if

```

4.4 Setting of Parameters

The setting of the two parameters f and n_c can affect the efficiency and quality of summarization. Let h be the height of the tree (root is at level 1). For simplicity, we always construct full trees and allow $n_c = f^h$ to vary from hundreds to thousands. For example, for $f = 15$, we can either use a two-layer tree, where the number of leaf entries $n_c = 225$, or a three-layer tree where $n_c = 3375$. In experiment, we show that a smaller f always results in faster summarization, but can undermine the quality of summarization when the clustering structure is changing. A small f may cause more imprecise mergence to happen in the second phase, since the less entries the lower level of precision is guaranteed for absorption. Larger f with the same height of tree will increase the cost – $O(dm f)$ in absorbing phase. On the other hand, larger f increases the ability adapting to the change of clustering structure since we can do more precise merging in the absorbing phase. To trade-off the performance and robustness, we can set the tree to be 2 ~ 3 layers, with $f = 10 \sim 20$.

4.5 Complexity of HE-Tree

The time complexity of constructing HE-Tree can be divided into two phases. In the growing phase, about f^h records are inserted into the tree and each record needs at most $O(hf)$ comparison to locate the target node. In the absorption phase, besides the cost of locating, each record needs an ACE-style mergence, which costs $O(dm f)$. Since f is usually a small value, e.g. $10 \sim 20$ and $h = 2$ or 3 in practice. Thus, the total cost is only dominated by the number of dimensions d and the average cardinality m of the dataset, i.e. the factor dm .

There are $O(f^h)$ nodes in the tree. Each leaf node needs approximately $O(fdm + f^2)$ space, where the summary table for each entry needs $O(dm)$ and the I_m table needs $O(f^2)$. Each internal node needs only $O(fdm)$, holding the summary tables and the pointers to the children nodes. Approximately, a HE-Tree needs $O((dm + f)f^{h+1})$ space. With fixed small f and h , again only the factor dm of the dataset determines the size of the tree. Except the datasets having very large dm , e.g. over 10k, HE-Tree usually needs small amount of memory.

5 Framework for Monitoring the Change of Clustering Structure

With the HE-Tree and the ACE algorithm, we can precisely monitor the change of clustering structure in the categorical data stream. The framework is illustrated in Figure 6.

1. The records from the data stream are inserted into the HE-Tree by order. Each insertion costs $O((h+dm)f)$;
2. At some time interval, the summary tables in the leaf nodes are dumped out. It only costs $O(dmn_c)$ bytes to store each of such snapshots;
3. ACE algorithm are performed on the snapshot as soon as it is dumped, the result of which generates a BkPlot. The cost is $O(dmn_c^2)$ [9].

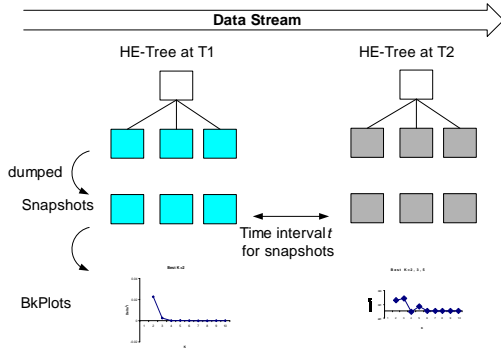


Figure 6: Framework for detecting change of clustering structure in categorical data streams.

Basically, the cost of first step restricts how many records the framework can monitor in unit time. As we have shown, f also affects the precision of summarization. There is a tradeoff between the precision and the capacity of monitoring system, tuned by the parameter f . The cost of third step affects how frequently we can generate a BkPlot. The time interval t in the second step is directly determined by the cost of generating BkPlot, i.e., t should be greater than $O(dmn_c^2)$. Reducing n_c allows more snapshots to be processed in unit time, and thus more details about the changes to be observed. But in practice, $n_c = 400 \sim 1000$ is enough to generate precise BkPlot, which means $O(dmn_c^2)$ is usually a small number and thus we can observe the BkPlots in a high frequency.

6 Experimental Results

The goal of the experiments is two-fold. 1) We investigate the parameter setting of HE-Tree and give the estimate of appropriate settings; 2) We want to show that HE-Tree summarization together with ACE clustering algorithm can provide high-quality monitoring result.

Datasets We construct a synthetic dataset DS1 with the following way, so that the clustering structure can be intuitively identified and manually labeled before running the experiments. The synthetic dataset has a two-layered clustering structure (Figure 7) with 30 attributes and N rows. It has four same-sized clusters in the top layer. Each cluster has random categorical values selected from $\{‘0’, ‘1’, ‘2’, ‘3’, ‘4’, ‘5’\}$ in some distinct set of attributes (the dark area in Figure 7), while the rest attributes are set

to ‘0’. Two of the four clusters also have clustering structure in the second layer. This synthetic data has clearly defined clustering structure, and each record in the dataset distinctly belongs to one cluster. This dataset is primarily used in exploring the effect of the parameters of HE-Tree to the precision of clustering result and the efficiency of summarization. And later, it is also used to demonstrate a sequence of monitoring results. We also use a real

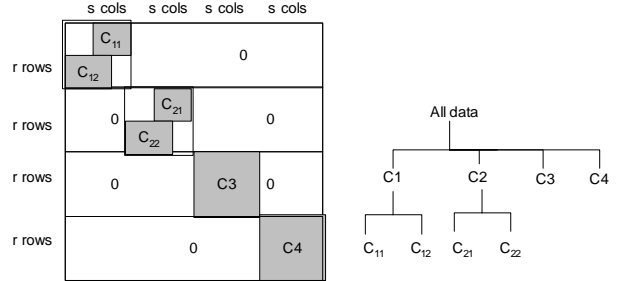


Figure 7: Clustering structure of DS1

dataset: “US Census 1990 Data ” in the experiment. This dataset is a discretized version of the raw census data, originally used by [22]. It can be found in UCI KDD Archive¹. Many of the less useful attributes in the original data set have been dropped, the few continuous variables have been discretized and the few discrete variables that have a large number of possible values have been collapsed to have fewer possible values. The total number of preserved attributes is 68. The first attribute is the sequence number, which is discarded in clustering. This dataset is very large, containing about 2 million records.

Error Rate The cluster labels in the synthetic dataset DS1 allow us to evaluate the quality of clustering result more accurately by using the *Error Rate* measure. Suppose the best K is identified. The Error Rate is defined based on the *confusion matrix*, where each element c_{ij} $1 \leq i, j \leq K$ represents the number of points from the labeled cluster j assigned to cluster i by the algorithm. Let $\{(1), (2), \dots, (K)\}$ be any permutation of sequence $\{1, 2, \dots, K\}$. There is a permutation that maximizes the number of consistent points m_c , which is defined as follows.

$$m_c = \max \left\{ \sum_{i=1}^K c_{i(i)}, \text{ for any } \{(1), (2), \dots, (K)\} \right\}$$

We define Error Rate as $1 - \frac{m_c}{N}$, N is the total number of points.

6.1 Parameter Setting for HE-Tree

For a full tree, the fan-out f of tree node and the height of the tree determine the tree structure. To simplify the investigation and maximize the quality of the summarization, we always use full trees in the experiment. Intuitively, for a fixed f , the higher tree (the larger h), the finer granularity of summarization will be delivered. However, most likely we care about only the clustering structures having less than 20 clusters. Therefore, a short tree, which generates less than one thousand sub-clusters, is enough for achieving high-quality BkPlot with ACE clustering algorithm. The experiments will be focused on the full short

¹<http://kdd.ics.uci.edu/>

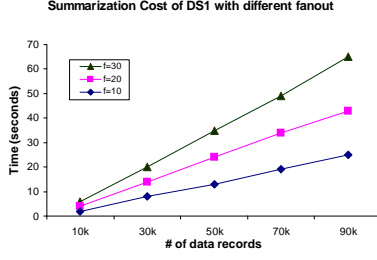


Figure 8: Cost of HE-Tree summarization with different fanout f .

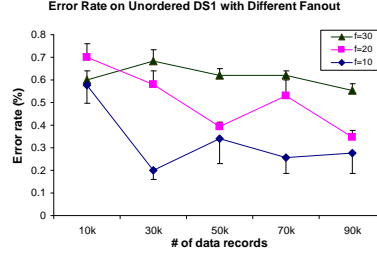


Figure 9: Error rate of ACE clustering result with HE-Tree summarization on randomly ordered records

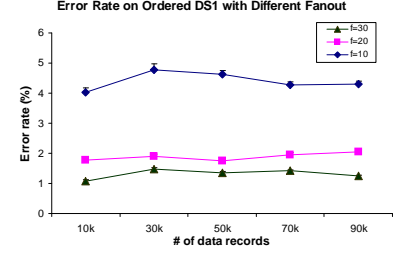


Figure 10: Error rate of ACE clustering result with HE-Tree summarization on ordered records

trees (e.g., $h = 2$) with varying fan-out f from 10 to 30. A set of datasets (20 datasets) in the same structure shown in Figure 7 are generated, the result is statistically based on the 20 runs.

Figure 8 shows the cost for HE-Tree summarization is linear and the cost also varies linearly according to f , which is consistent with our analysis. Figure 9 shows the effect of different summarization structures to the quality of final clustering result for “Unordered DS1”. Unordered DS1 randomly stores the records from different clusters, i.e., there is little change of clustering structure in processing the entire data stream. The result shows some variances between the error rates for different f , but overall the error rates are similar and low.

“Ordered DS1” shows a more interesting scenario, where clustering structure dramatically changes when time goes by. In such situations, f may significantly affect the quality of monitoring. Figure 10 shows the result of sequentially processing the clusters C_{11} to C_4 . A tree with larger f seems more adaptive to the change of clustering structure. The reason that HE-Tree is more sensitive to the setting of f when the clustering structure changes dramatically can be understood as follows. The initial records from the same cluster already occupy the slots in the growing stage. When a new cluster emerges, since there is no entry in the tree belonging to the new cluster, new slots are given to the new records by merging other similar entries, or, some initial records may be absorbed to other clusters by small chance, which causes the error. It shows that increasing f from 10 to 20 can considerably reduce the error, but $f = 30$ will not significantly improve the result of $f = 20$. Balanced with the time cost and the robustness, $f = 20$ seems the best for efficiently adapting the change of structure.

6.2 Robustness of BkPlots by HE-Tree/ACE

In this experiment, we want to compare the accuracy of BkPlots generated by ACE on small sample set and by HE-Tree/ACE on large stream data. We run the experiment on both the synthetic data and the real US Census data. The small sample size is set to 500 for ACE, and large samples sizes are 10K and 30K. The sample sets are uniformly drawn from the original dataset, therefore, they are supposed to have the same clustering structure.

Figure 11 for DS1 shows all of the three BkPlots can identify the primary best Ks: 4 and 6, while a little noise appears at $K=2$ when the sample size is large. All BkPlots for Census data (Figure 12) strongly suggest the best $K=2$, while $K=7$ is probably another candidate. The result confirms that HE-Tree summarization can preserve the primary

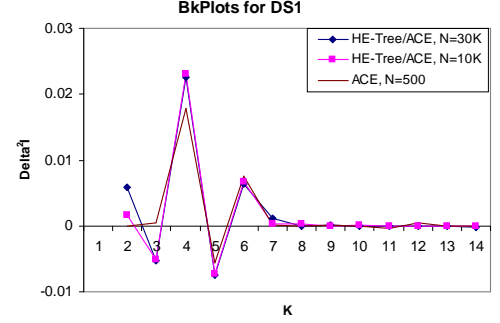


Figure 11: BkPlots for DS1.

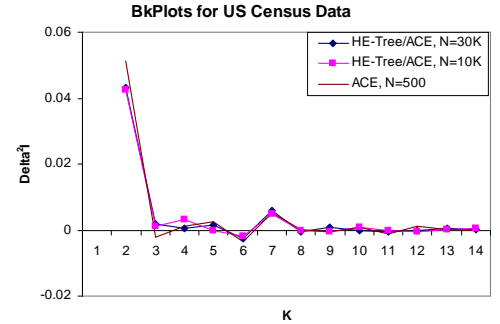


Figure 12: BkPlots for Census.

clustering structure and thus HE-Tree combined with ACE method is a robust approach for monitoring the change of clustering structure.

6.3 Monitoring the Changes

We also demonstrate the progressive monitoring results of the two data streams: DS1-stream and Census-stream. The DS1-stream simulates the 4/6-cluster structure shown in Figure 7. The clusters enter the stream in the sequence of C_{11} , C_{12} , C_{21} , C_{22} , C_3 , and C_4 . Each of the small clusters have 5K records and each of the large clusters have 10K records. Snapshots are saved at $N=10K$, $20K$, and $30K$, respectively.

The progressive results for DS1-stream in Figure 13 clearly identify the change of clustering structure. At $T1:N=10K$, C_{11} and C_{12} have been present at the stream, thus two clusters are identified. At $T2:N=20K$, C_{21} and C_{22} emerge and the two-layer structure is identified (the best $K = 2, 4$). At $T3:N=30K$, C_3 appears, and the BkPlot detects that the primary two-layer structure is changed to $K=3, 5$, while the BkPlot also suggest an additional layer at $K=2$, which consists two cluster (C_{11}, C_{12}, C_{21} and C_{22})

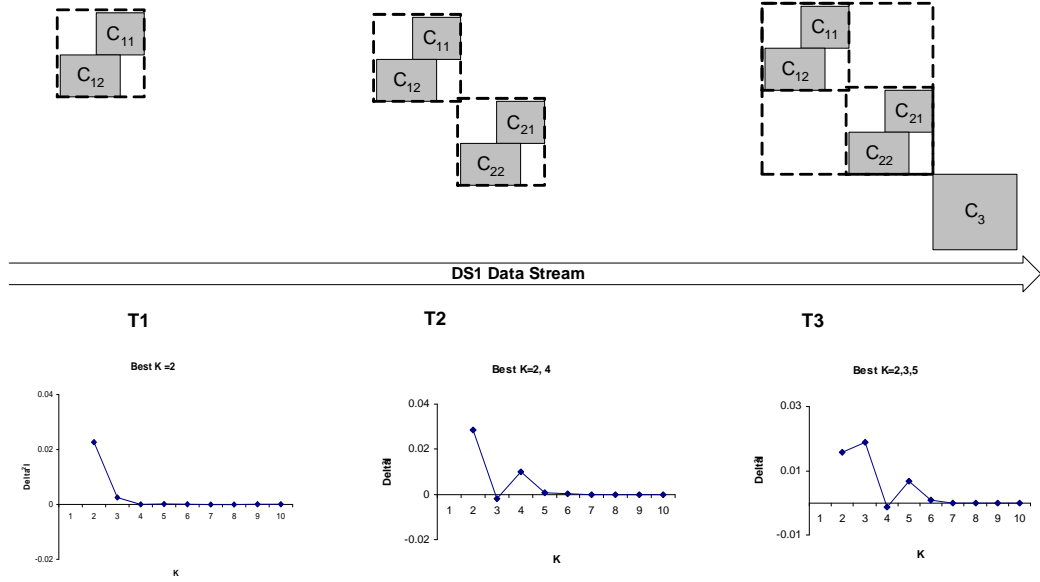


Figure 13: Monitoring DS1-stream.

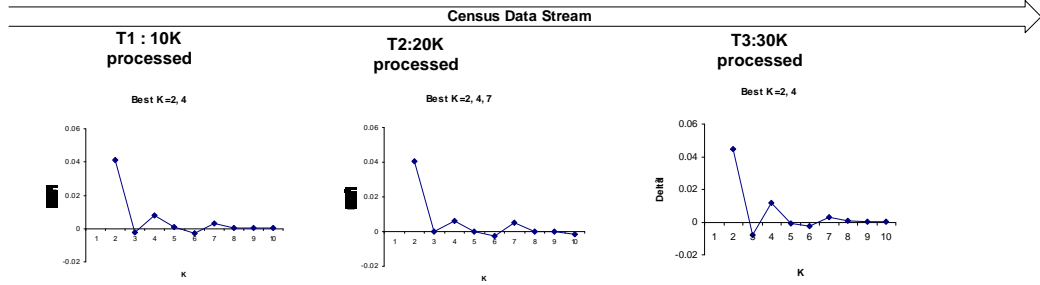


Figure 14: Monitoring Census-stream.

and (C_3).

In Census-stream, the primary clustering structure (best $K=2$) keeps unchanged. However, there are new sub-level clusters emerging. $K=4$ emerges as a candidate at $N=10K$, and it becomes clear at $30K$. On the contrary, $K=7$ decays at $N=30K$ and is finally identified as a noisy sub-clustering structure.

7 Related Work

While many numerical clustering algorithms [19, 20] have been published, only a handful of categorical clustering algorithms appear in literature. Although it is unnatural to define a distance function between categorical data or to use the statistical center (the mean) of a group of categorical items, there are some algorithms, for example, K-Modes [18] algorithm and ROCK [16] algorithm, trying to fit the traditional clustering methods into categorical data. However, since the numerical similarity/distance function may not describe the categorical properties properly and intuitively, it leaves little confidence to the clustering result. CACTUS [13] also partly adopts the linkage idea used in ROCK.

Gibson et al. introduced STIRR [14], an iterative algorithm based on non-linear dynamical systems. STIRR represents each attribute value as a weighted vertex in a graph. Starting with the initial conditions, the system is iterated until a “fixed point” is reached. When the fixed point is reached, the weights in one or more of the “basins” iso-

late two groups of attribute values on each attribute. Even though they proved this approach works for some experimental datasets having two partitions, the user may hesitate in using it due to the complicated and not intuitive working mechanism.

Coolcat [3] is kind of similar to KModes. However, Coolcat assigns the item to the cluster that minimizes the expected entropy. Considering the cluster centers may shift, a number of worst-fitted points will be re-clustered after a batch. Li et al [21] proposed a Monte-Carlo method to minimize the expected entropy, which is slower than Coolcat but can be more likely to achieve the sub-optimal results. Cross Association [8] tries using MDL to partition boolean matrix along row direction and column direction at the same time. In fact, MDL is equivalent to entropy criterion as [21] shows. Some closely related work also borrows concepts from information theory, including Co-clustering [12], Information Bottleneck [23] and LIMBO [2]. The results all show that the entropy related concepts work very well for categorical data.

Clustering data streams becomes one of the important technique for analyzing the data streams [15]. In [1], a framework is proposed for clustering evolving data streams, which mainly concerns numerical data. Clustering categorical data stream was first addressed by Coolcat[3], but no more related issues such as detecting the change of clustering structure are addressed yet. [4] addresses that detecting change in data stream has various applications,

and we propose that monitoring the change of clustering structure is also very useful.

8 Conclusion

In this paper, we address the problem of detecting the change of clustering structure in categorical data streams, with a novel framework. The framework is based on the BkPlot method, which was applied in best K detection in general categorical clustering, and focuses on the Hierarchical Entropy Tree (HE-Tree) summarization structure. HE-Tree is designed as a memory-efficient structure – the tree is usually a short tree (height = 2 or 3) with small number of leaf nodes. HE-Tree is also efficient in drawing the entropy information from the data stream and preserving it in a number of coarse clusters. Snapshots can be made on the HE-Tree in certain time interval in order to observe the change of clustering structure. ACE clustering algorithm is then performed on the snapshots to generate precise BkPlots, with which we can easily identify whether and how the clustering structure in the stream is changed.

Future work includes the finer structure for preserving and observing more information in the categorical data stream. In fact, the difference between the snapshots should also provide some interesting things. The application of the proposed framework will also be investigated in various time-critical areas that generates categorical data streams.

References

- [1] AGGARWAL, C. C., HAN, J., WANG, J., AND YU, P. S. A framework for clustering evolving data streams. *Proc. of Very Large Databases Conference (VLDB)* (2004).
- [2] ANDRITSOS, P., TSAPARAS, P., MILLER, R. J., AND SEVCIK, K. C. Limbo:scalable clustering of categorical data. *Proc. of Intl. Conf. on Extending Database Technology (EDBT)* (2004).
- [3] BARBARA, D., LI, Y., AND COUTO, J. Coolcat: an entropy-based algorithm for categorical clustering. *Proc. of ACM Conf. on Information and Knowledge Mgt. (CIKM)* (2002).
- [4] BEN-DAVID, S., GEHRKE, J., AND KIFER, D. Detecting change in data stream. *Proc. of Very Large Databases Conference (VLDB)* (2004).
- [5] BOCK, H. Probabilistic aspects in cluster analysis. *Conceptual and Numerical Analysis of Data* (1989).
- [6] BRAND, M. An entropic estimator for structure discovery. In *Proc. Of Neural Information Processing Systems (NIPS)* (1998), pp. 723–729.
- [7] CELEUX, G., AND GOVAERT, G. Clustering criteria for discrete data and latent class models. *Journal of Classification* (1991).
- [8] CHAKRABARTI, D., PAPADIMITRIOU, S., MODHA, D. S., AND FALOUTSOS, C. Fully automatic cross-associations. *Proc. of ACM SIGKDD Conference* (2004).
- [9] CHEN, K., AND LIU, L. The “best k” for entropy-based categorical clustering. *Proc. of Intl. Conf. on Scientific and Statistical Database Management (SS-DBM)* (2005).
- [10] CHENG, C. H., FU, A. W.-C., AND ZHANG, Y. Entropy-based subspace clustering for mining numerical data. *Proc. of ACM SIGKDD Conference* (1999).
- [11] COVER, T., AND THOMAS, J. *Elements of Information Theory*. Wiley, 1991.
- [12] DHILLON, I. S., MELLELA, S., AND MODHA, D. S. Information-theoretic co-clustering. *Proc. of ACM SIGKDD Conference* (2003).
- [13] GANTI, V., GEHRKE, J., AND RAMAKRISHNAN, R. CACTUS-clustering categorical data using summaries. *Proc. of ACM SIGKDD Conference* (1999).
- [14] GIBSON, D., KLEINBERG, J., AND RAGHAVAN, P. Clustering categorical data: An approach based on dynamical systems. *Proc. of Very Large Databases Conference (VLDB)* 8, 3–4 (2000), 222–236.
- [15] GUHA, S., MEYERSON, A., MISHRA, N., MOTWANI, R., AND O’CALLAGHAN, L. Clustering data streams: Theory and practice. *IEEE Trans. on Knowledge and Data Eng.* 15 (2003).
- [16] GUHA, S., RASTOGI, R., AND SHIM, K. ROCK: A robust clustering algorithm for categorical attributes. *Proc. of IEEE Intl. Conf. on Data Eng. (ICDE)* (1999).
- [17] HALKIDI, M., BATISTAKIS, Y., AND VAZIRGIAN-NIS, M. Cluster validity methods: Part I and II. *SIGMOD Record* 31, 2 (2002), 40–45.
- [18] HUANG, Z. A fast clustering algorithm to cluster very large categorical data sets in data mining. *Workshop on Research Issues on Data Mining and Knowledge Discovery* (1997).
- [19] JAIN, A. K., AND DUBES, R. C. *Algorithms for Clustering Data*. Prentice hall, 1988.
- [20] JAIN, A. K., AND DUBES, R. C. Data clustering: A review. *ACM Computing Surveys* 31 (1999).
- [21] LI, T., MA, S., AND OGIHARA, M. Entropy-based criterion in categorical clustering. *Proc. of Intl. Conf. on Machine Learning (ICML)* (2004).
- [22] MEEK, C., THIESSON, B., AND HECKERMAN, D. The learning-curve sampling method applied to model-based clustering. *Journal of Machine Learning Research* 2 (2002), 397–418.
- [23] TISHBY, N., PEREIRA, F. C., AND BIALEK, W. The information bottleneck method. *Proc. of the 37-th Annual Allerton Conference on Communication, Control and Computing* (1999).
- [24] YANG, Y., AND PEDERSEN, J. O. A comparative study on feature selection in text categorization. *Proceedings of ICML-97, 14th International Conference on Machine Learning* (1997), 412–420.